APPENDIX D

```
  1 #
  2 # procOrder.pl (c) 1998,1999 E-Dialog, Inc. All Rights Reserved.
  3 #
  4 # PURPOSE DISCUSSION
  5 #
  6 # program accepts a raw email stream from MS Exchange Server via export to ADO-compliant datastore
  7 # then uses email body preprocessing and extraction rules - tailored per campaign in ./cf/ - to construct order reports.
  8 #
  9 #       cf files are campaign optimized:
 10 #
 11 #       mailingDBLayout.cf                  -> field mappings for a campaign's master mailing table
 12 #       messageBodyDataLayout.cf            -> DSNs, field mappings, critical rule designations, order-designation bounds, synonymn table
 13 #       messageBodyIndicateAddressChange.cf -> rulesets for indicating ADDRESS CHANGES
 14 #       messageBodyIndicateBuyAll.cf        -> rulesets for indicating BUY ALL state
 15 #       messageBodyPreProcRules.cf          -> body text preprocessor configuration and rulesets
 16 #
 17 # program produces diagnostic and order-entry bound output files and logs outcomes of all ruleset evaluations leading to
 18 # preparation of those output files, which include:
 19 #
 20 #       <$ARGV[1]>              -> summary order report file (designed for data entry, faxing, emailing to fulfillment)
 21 #       EXC_<...>               -> (companion) summary exception report (same format as s. order report with annotations to indicate rule failures for man
    ual review)
 22 #       OUT_<...>               -> detailed order stream file (all fields discretely mapped and normalized for input into batch postprocessor)
 23 #       EXC_DETAIL_<...>        -> (companion) ..exceptions..discrete fields, etc.
 24 #       EXC_BODY_<...>          -> annotated diagnostic report containing all message bodies - sans original message - for aid in resolving exceptions man
    ually
 25 #
 26 # output files are (default) tab-delimited text
 27 #
 28 # PROGRAM REQUIRES
 29 #
 30 #       Perl 5.003 interpreter later compiled for WIN32
 31 #       MS ADO components 1.5 (2.0+ recommended)
 32 #
 33 # RELEASE HISTORY
 34 #
 35 # 981020AE: 0.5         first usage: to extract carol10 order reports
 36 # 981204AE: 0.9         upgrade to handle carol12, multiple item selections
 37 # 981208AE: 1.1         improve flagging
 38 # 981211AE: 1.5         adapt to extract IDEAS@WORK add/changes
 39 # 981222AE: 2.3.1       fix bugs in EXC_BODY reporting / enhance it
 40 # 981222AE: 2.4         add cfPATH support
 41 # 981223AE: 2.4.1       cfPath logging; fix synonymn (SUBJECT TOKEN) lookup <trailing \s*>
 42 # 981223AE: 2.5         reorder reporting fields
 43 # 981224AE: 2.5.1       reorder reporting fields
 44 # 981225AE: 2.5.2       update SHIPPING_ADDRESS fields to reflect presence of T,B & S diagnostic fields
 45 # 981226AE: 2.6         code review/dox; update cf to require MAILINGID
 46 # 990106AE: 2.6.5       standardize on LOG ID for use in address change
 47 # 990129ae: 2.7         add support for separate billing, shipping addresses
 48 # 990201ae: 3.0         add record-recovery via ADO -> mailing database lookups
 49 # 990202ae: 3.0.1       record-update ADO -> mailing database
 50 # 990202ae: 3.0.2       fix leading digits bug in getDateTimeStamp ftn
 51 # 990203ae: 3.1         fix MDB query bugs; introduce detection of SELECT && UPDATE SUCCESS; verify decision-tree based UPDATES
 52 # 990205ae: 3.1.1       fix address3 bug; provide regression compatibility for /ADDCHANGE
 53 # 990206ae: 3.1.2       throw .noTok as a HARD_EXC
 54
 55 ##################### INCL #####################
 56
 57 use Win32::OLE;
 58
 59 ##################### ENV  #####################
 60
 61 $GrequiredARG   = 4;
 62 $Grevision      = '3.1.2';
 63
 64 if(scalar(@ARGV) < $GrequiredARG) { die "($Grevision) Usage: procOrder.pl [userName] [reportName] [cfPATH <campaignMnemonic>] [ <buyToken> | /multi | /add
    Change ]\n"; }
 65
 66 $Guser                          = $ARGV[0];
```

```
 67 $GreportFile                    = $ARGV[1];
 68 $GcfPath                        = $ARGV[2];
 69 $Gsubj_buyToken                 = $ARGV[3];
 70 $GexcReportFile                 = 'EXC_'.$GreportFile;
 71 $GexcBODYReportFile             = 'EXC_BODY_'.$GreportFile;
 72 $GoutFile                       = 'OUT_'.$GreportFile;
 73 $GexceptionsFile                = 'EXC_DETAIL_'.$GreportFile;
 74
 75 if($ARGV[4] == 1) { $GDEBUG = 1; }      else {$GDEBUG = 0;}
 76 if($ARGV[4] == 2) { $LGDEBUG_FTN = 1; } else {$LGDEBUG_FTN = 0; }
 77
 78 ##################### DEFN #######################
 79
 80 ## MISC GLOBALS ##
 81
 82 $Gplatform                       = 'WIN';
 83 $GeventType                      = 'INFO';
 84 $basePath                        = '.';
 85 $GfieldDelimiter                 = "\t";
 86 $Gspace                          = ' ';
 87 $Gnewline                        = "\n";
 88 $Gcomma                          = ',';
 89 $Gpipe                           = '|';
 90 $eventRequiresFUP                = '';
 91 $GmultiItemOrderSwitch           = '/MULTI';
 92 $GaddChangeSwitch                = '/ADDCHANGE';
 93
 94 ############# OUTPUT FILE LAYOUTS ###############
 95
 96 ## DO NOT change alpha ordering of lowercase field names (e_ .. t_)
 97 ## This is vital in the BODY field layout defn file since the rules reference the letter prefixes
 98
 99 ## reportFile (note: EXC_reportFileName also written from this template)
100
101         %GreportFileRecord = (
102
103                 a_LID                   => '_e_$thisLogKey',
104                 ab_LINE                 => '<fill-in>',
105                 b_RID                   => '_e_$thisRecordID',
106                 c_THIS_SUBJECT          => '_e_$thisSubject',
107                 d_THIS_BODY_BOTTOM      => '_e_$thisBodyBottom',
108                 e_THIS_BODY_TOP         => '_e_$thisBodyTop',
109                 ia_FUP_REQUIRED         => '_e_$eventRequiresFUP',
110                 ib_LOCATION_BUY_TOKEN   => '_e_$thisBuyStatus',
111                 j_REVIEW_ACTION_TAKEN   => '<review>',
112                 za_SELECTED_ITEMS       => '_e_$thisItemsSelectedStack',
113                 zd_PRIORITY             => '_e_$thisMailingID',
114                 zf_CUSTNO               => '_e_($thisCustNo==-1 || $thisCustNo == 0)?"":$thisCustNo',
115                 zi_LISTID               => '_e_($thisListID==-1)?"":$thisListID',
116                 zm_FROM_ADDRESS         => '_e_$thisFromAddress',
117                 zp_SPECIAL_INSTRUCTIONS => '<none>',
118                 zr_PHONE                => '_e_$results{\'r_bphone\'}',
119                 zx_BILLING_ADDRESS      => '_e_$thisBillingAddressBlock',
120                 zy_SHIPPING_ADDRESS     => '_e_$thisShippingAddressBlock',
121                 zz_MDB_BILLING_ADDRESS  => '_e_$thisMDBbillingAddressBlock',
122         );
123
124
125 ## outFile
126
127         %GoutFileRecord = (
128
129                 a_LID                   => '_e_$thisLogKey',
130                 aa_RID                  => '_e_$thisRecordID',
131                 ab_THIS_SUBJECT         => '_e_$thisSubject',
132                 ac_THIS_BODY_BOTTOM     => '_e_$thisBodyBottom',
133                 ad_THIS_BODY_TOP        => '_e_$thisBodyTop',
134                 b_LISTID                => '_e_$thisListID',
135                 bb_CUSTNO               => '_e_$thisCustNo',
```

```
136                 c_PRIORITY              => '_e_$thisMailingID',
137
138                 e_bfirst                => '_e_$results{\'e_bfirst\'}',
139                 f_bmiddle               => '_e_$results{\'f_bmiddle\'}',
140                 g_blast                 => '_e_$results{\'g_blast\'}',
141                 h_btitle                => '_e_$results{\'h_btitle\'}',
142                 i_bcompany              => '_e_$results{\'i_bcompany\'}',
143                 j_bdepartment           => '_e_$results{\'j_bdepartment\'}',
144                 k_baddress1             => '_e_$results{\'k_baddress1\'}',
145                 l_baddress2             => '_e_$results{\'l_baddress2\'}',
146                 m_baddress3             => '_e_$results{\'m_baddress3\'}',
147                 n_bcity                 => '_e_$results{\'n_bcity\'}',
148                 o_bstate                => '_e_$results{\'o_bstate\'}',
149                 p_bpostal               => '_e_$results{\'p_bpostal\'}',
150                 q_bcountry              => '_e_$results{\'q_bcountry\'}',
151                 r_bphone                => '_e_$results{\'r_bphone\'}',
152                 s_bfax                  => '_e_$results{\'s_bfax\'}',
153                 t_bemail                => '_e_$results{\'t_bemail\'}',
154
155                 ue_sfirst               => '_e_$results{\'ue_sfirst\'}',
156                 uf_smiddle              => '_e_$results{\'uf_smiddle\'}',
157                 ug_slast                => '_e_$results{\'ug_slast\'}',
158                 uh_stitle               => '_e_$results{\'uh_stitle\'}',
159                 ui_scompany             => '_e_$results{\'ui_scompany\'}',
160                 uj_sdepartment          => '_e_$results{\'uj_sdepartment\'}',
161                 uk_saddress1            => '_e_$results{\'uk_saddress1\'}',
162                 ul_saddress2            => '_e_$results{\'ul_saddress2\'}',
163                 um_saddress3            => '_e_$results{\'um_saddress3\'}',
164                 un_scity                => '_e_$results{\'un_scity\'}',
165                 uo_sstate               => '_e_$results{\'uo_sstate\'}',
166                 up_spostal              => '_e_$results{\'up_spostal\'}',
167                 uq_scountry             => '_e_$results{\'uq_scountry\'}',
168                 ur_sphone               => '_e_$results{\'ur_sphone\'}',
169                 us_sfax                 => '_e_$results{\'us_sfax\'}',
170                 ut_semail               => '_e_$results{\'ut_semail\'}',
171
172                 ux_FROM_ADDRESS         => '_e_$thisFromAddress',
173                 uy_LOCATION_BUY_TOKEN   => '_e_$thisBuyStatus',
174                 v_SUBJECT               => '_e_$thisSubject',
175                 z_FUP_REQUIRED          => '_e_$eventRequiresFUP',
176                 zz_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
177         );
178
179 ## exceptionsFile - BODY only
180
181         %GexceptionsBODYFileRecord = (
182
183                 a_LID                   => '_e_("================================================================================================================
    ===========\n>>".$thisLogKey."<<")',
184                 aa_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
185                 ab_EXCEPT_FLAGS         => '_e_$eventRequiresFUP',
186                 b_RID                   => '_e_("<<".$thisRecordID.">>")',
187                 u_FROM_ADDRESS          => '_e_("||".$thisFromAddress."||")',
188                 v_SUBJECT               => '_e_("||".$thisSubject."||\n================================================================================================
    ==================================")',
189                 w_BODY_BELOW            => '_e_$thisFullNumberedBody',                  # full body
190
191         );
192
193 ## activityLogFile
194
195         %GactivityLogRecord = (
196
197                 a_LID                   => '_e_$thisLogKey',
198                 b_user                  => '_e_$Guser."/".$GcfPath',
199                 c_dateStamp             => '_e_&getDateStamp(\'WIN\',$LGDEBUG_FTN)',
200                 e_RID                   => '_e_$thisRecordID',
201                 f_clientName            => '_e_$GclientName',
202                 g_eventType             => '_e_$eventType',
```

```
203                 h_eventDesc             => '_e_$eventDesc',
204                 z_FUP_REQUIRED          => '_e_$eventRequiresFUP',
205         );
206
207
208 ################### DATA ####################
209
210 &getKeyPos_Outfile(\%GoutFileRecord);          # identify the fieldwise position of fields in the outfile for canonicalization (use EXC file as model)
211
212 ##print "--- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
213
214 $GDSN                           = 'dsnOrderProcessing';
215
216 if ($Gplatform ne 'WIN') {$relativePath = '/logs/'} else { $relativePath = '\\logs\\';}
217
218 $GactivityLogFileName           = $basePath.$relativePath.'activityLog.tab';
219 $GactivityLogKeysFileName       = $basePath.$relativePath.'activityLogKeys.tab';
220
221
222 if ($Gplatform ne 'WIN') {$relativePath = '/db/'} else { $relativePath = '\\db\\';}
223
224 $GexcReportDBkeysFileName       = $basePath.$relativePath.'excReportDBkeys.tab';
225 $GexceptionsDBkeysFileName      = $basePath.$relativePath.'excDetailDBkeys.tab';
226 $GexceptionsBODYDBkeysFileName  = $basePath.$relativePath.'excBodyDBkeys.tab';
227 $GoutFileDBkeysFileName         = $basePath.$relativePath.'outFileDBkeys.tab';
228 $GreportFileDBkeysFileName      = $basePath.$relativePath.'reportFileDBkeys.tab';
229
230 if ($Gplatform ne 'WIN') {$relativePath = 'cf'.'/'.$GcfPath} else { $relativePath = '\\cf\\'."$GcfPath\\";}
231
232 $GbodyDataLayoutFileName        = $basePath.$relativePath.'messageBodyDataLayout.cf';
233 $GbodyPreProcRulesFileName      = $basePath.$relativePath.'messageBodyPreProcRules.cf';
234 $GbodyBuyAllFileName            = $basePath.$relativePath.'messageBodyIndicateBuyAll.cf';
235 $GbodyAddressChangeFileName     = $basePath.$relativePath.'messageBodyIndicateAddressChange.cf';
236 $GmailingDBLayoutFileName       = $basePath.$relativePath.'mailingDBLayout.cf';
237
238 ################################################
239 ################### MAIN ####################
240 ################################################
241
242 $MODE                   = '';
243 $MULTI_BUY_ON           = 0;
244 $ADD_CHANGE_ON          = 0;
245 $Gsubj_buyToken         = uc($Gsubj_buyToken);
246
247 $MODE = "<TOKEN = |$Gsubj_buyToken|>";
248 if($Gsubj_buyToken =~ /\/MULTI/)        { $MULTI_BUY_ON = 1; $MODE .= '<MULTI BUY ON>';}
249 if($Gsubj_buyToken =~ /\/ADDCHANGE/)    { $ADD_CHANGE_ON = 1; $MODE .= '<ADDCHANGE ON>'; }
250
251 print "[procOrder R.$Grevision] $MODE\n".($GDEBUG?"\n<DEBUG: $GDEBUG:$LGDEBUG_FTN>\n\n":"\n");
252
253 ## open CF files
254
255 print "** Opening: $GbodyPreProcRulesFileName...\n";
256         @GPreProcRules                                  = &loadFile_SCALAR($GbodyPreProcRulesFileName);
257
258 print "** Opening: $GbodyBuyAllFileName...\n";
259         @GBuyAllIndicators                              = &loadFile_SCALAR($GbodyBuyAllFileName);
260
261 print "** Opening: $GbodyAddressChangeFileName...\n";
262         @GAddressChangeIndicators                       = &loadFile_SCALAR($GbodyAddressChangeFileName);
263
264 print "** Opening: $GbodyDataLayoutFileName...\n";
265         %LAYOUT                                         = &HASHloadFieldMap($GbodyDataLayoutFileName,$LGDEBUG_FTN);
266
267 unless($ADD_CHANGE_ON) {
268
269         print "** Opening: $GmailingDBLayoutFileName...\n";
270         %MDBLAYOUT = &HASHloadFieldMap($GmailingDBLayoutFileName,$LGDEBUG_FTN);
271
```

```
272 }
273
274 ## open I/O
275
276 if (!open(FH_log, ">>$GactivityLogFileName"))          {$eventType = 'ABORT'; $eventDesc = "MAIN::Cannot open GactivityLogFileName: \[$fileName\]"; &logE
    vent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
277 if (!open(FH_out, ">$GoutFile"))                     {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GoutFile: \[$GoutFile\]"; &logEvent(\*FH_log
    , $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
278 if (!open(FH_report, ">$GreportFile"))                {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GreportFile: \[$GreportFile\]"; &logEvent(\*
    FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
279 if (!open(FH_excReport, ">$GexcReportFile"))          {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excReportFile: \[$GexcReportFile\]"; &logEve
    nt(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
280 if (!open(FH_excBODYReport, ">$GexcBODYReportFile"))  {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excBODYReportFile: \[$GexcBODYReportFile\]";
     &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
281 if (!open(FH_exc, ">$GexceptionsFile"))               {$eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GexceptionsFile: \[$GexceptionsFile\]"; &log
    Event(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
282
283 ## load %results fieldName keys from layout cf file (now in %LAYOUT)
284
285 @fieldKeys              = undef;
286 @fieldKeys              = &getFieldKeys(\%LAYOUT);
287
288         ##print "--debug--\n"; foreach(@fieldKeys) { print "**".$_."| \n"; }
289
290 $numberRequiredFields   = scalar(@fieldKeys);
291
292 if(!defined($LAYOUT{'orderProcessingDSN'})) { $LAYOUT{'orderProcessingDSN'} = $GDSN; }
293
294 print "** Loading daily order data source: $LAYOUT{'orderProcessingDSN'}...\n";
295 @DB                     = &read_DSNtoAOH($LAYOUT{'orderProcessingDSN'},$LGDEBUG_FTN);
296
297 $totalRecords           = $#DB +1;
298 $GclientName            = $LAYOUT{'thisClientName'};
299
300 ## startup OK event
301
302 $eventType = '_x_INFO'; $eventDesc = "procOrder R.$Grevision START OK - USER: $Guser/$GcfPath CLIENT: $GclientName  opDSN ($totalRecords): $LAYOUT{'orderP
    rocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thisMailingTable'}  ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\
    *FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
303
304 print "\n\n** Status...\n\n";
305 print "USER:\t\t$Guser \n";
306 print "CLIENT:\t\t$GclientName\n";
307 print "cfPATH:\t\t$GcfPath\n\n";
308 print "TOTAL RECORDS \t\t -> \"$LAYOUT{'orderProcessingDSN'}\":  $totalRecords\n";
309 unless($ADD_CHANGE_ON) { print "MAILING TABLE SOURCE \t -> \"$LAYOUT{'thisMailingDSN'} -> $LAYOUT{'thisMailingTable'}\"\n\n"; }
310 print "\n(Note: if this information is not correct, hit <CNTRL> - C to abort and correct <you have 5 seconds>)\n\n"; sleep 5;
311
312 print "** Begin processing...\n\n";
313
314 ## flag duplicate e-mail addresses in order stream
315
316 %dupsMap = &checkDupsReturnMap(\@DB,$GDEBUG);
317
318 ###################
319 #### main loop ####
320 ###################
321
322 $excpCount              = 0;
323 $warnCount              = 0;
324 $outFileCount           = 0;
325 $exceptionsFileCount    = 0;
326 $excReportFileCount     = 0;
327 $reportFileCount        = 0;
328 $FUPcount               = 0;
329 $reportLineCount        = 0;
330
331 for $mainCount (0 .. $#DB) {
332
```

```
333 $thisRecordID                   = $mainCount+1;
334 $eventRequiresFUP               = '';
335 $RECORD_VALID                   = 1;
336 $thisBuyStatus                  = '';
337 $thisShippingAddressBlock       = undef;
338 $thisBillingAddressBlock        = undef;
339 $thisMDBbillingAddressBlock     = undef;
340 $thisFullNumberedBody           = "\n";
341
342 ## get relevant fields, normalized, this record
343
344 $thisFromAddress        = &normalizeSpacing(lc($DB[$mainCount]{'FromAddress'}));
345 $thisFromAddress        =~ s/[\t\r]/$Gspace/g;
346
347 $thisFromName           = &normalizeSpacing($DB[$mainCount]{'FromName'});
348 $thisFromName           =~ s/[\t\r]/$Gspace/g;
349
350 $thisSubject            = &normalizeSpacing(uc($DB[$mainCount]{'Subject'}));
351 $thisSubject            =~ s/[\t\r]/$Gspace/g;
352
353 $thisBody               = uc($DB[$mainCount]{'Body'});
354 $thisBody               =~ s/[\t\r]/$Gspace/g;
355
356 ## split BODY into discrete lines
357
358 @BODYlines              = undef;
359 @topLines               = undef;
360 @bottomLines            = undef;
361
362 push(@BODYlines, $1) while ($thisBody =~ s/^([^\012\015]*)(\012\015?|\015\012?)//);
363
364 ## skip lines that are the quoted BODY unless they're the order form -- example delimiters:
365 ## $BODYstartToken = 'HARVARD BUSINESS SCHOOL PUBLISHING';
366 ## $BODYendToken = 'FIRST NAME\:';
367
368 $BODYstartToken         = $LAYOUT{'startDelimiter'};
369 $BODYendToken           = $LAYOUT{'endDelimiter'};
370 $BODYstartToken         =~ s/[\t\n\r]//g;
371 $BODYendToken           =~ s/[\t\n\r]//g;
372 $startToken_ON          = 0;
373 $endToken_ON            = 0;
374 $thisBody               = undef;
375 $thisBodyTop            = undef;
376 $thisBodyBottom         = undef;
377 $lastLineExcluded       = undef;
378
379 for($count = 0; $count <= $#BODYlines; $count++) {
380
381         $line                   = $BODYlines[$count];
382         $thisFullNumberedBody   .= ('['.$count."] \t".$line."\n");
383
384         unless($startToken_ON || $line =~ /^\>+/) {unless ($line !~ /\S+?/) {$thisBodyTop .= ($line.$Gpipe)}; push(@topLines,('['.$count.']  |'.$line.'|')
    ); }
385
386                 if(($line =~ /$BODYstartToken/) .. ($line =~ /$BODYendToken/)) {
387
388                         if(!$startToken_ON && ($line =~ /$BODYstartToken/))     {$startToken_ON = 1;}
389                         if(!$endToken_ON && ($line =~ /$BODYendToken/))         {$endToken_ON = 1; $thisBody .= ($line.$Gnewline); unless($line =~ /^\>+/
    || $line !~ /\S+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe)} }
390
391                 next;
392                 }
393
394         ## ignore bottom lines if part of the original -> indicators:  ">" || "[" || "]"
395
396         if($startToken_ON && $endToken_ON) { unless($line =~ /^\>+/ || $line !~ /\S+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe); push(@bo
    ttomLines, ('['.$count.']  |'.$line.'|'));} }
397         $thisBody .= ($line.$Gnewline);
```

```
399
400 $thisBodyTop    =~ s/[\t\r]/__/g;
401 $thisBodyBottom =~ s/[\t\r]/__/g;
402
403                 if($GDEBUG) { print "\n\n\n** DEBUG__THISBODY=".$thisBody."\n\n\n";}
404
405 ## log/print warning if cannot find FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID || if bad match yields the same for CUSTNO|LISTID
406
407         $thisCustNo             =  -1;
408         $thisListID             =  -1;
409         $thisMailingID          =  -1;
410         $thisDialogID           =  -1;
411         $LISTCID_EXCEPT_ON      =   0;
412         $MAILINGID_EXCEPT_ON    =   0;
413
414         if($thisBody !~ /\[\[([^|]*?)\|([^|]*?)\|([^|]*?)\|?([^|]*?)\]\]/) {
415
416                 $LISTCID_EXCEPT_ON = 1;
417
418                 $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   UNMATCHED FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID";
    &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
419                 $eventType = 'INFO'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   >>> ATTEMPTING SUBJECT LINE MAILING ID RECOVERY >>>"; &logEvent(
    \*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
420
421 ## test to see if there is a cf file synonymn (**,++, etc) for the MAILING ID which can be recovered in the absence of the footer token
422
423                 if($thisMailingID == -1) { $thisMailingID = &subjectSynonymnLookup($thisSubject,\%LAYOUT); }
424                 if($thisMailingID == -1) {
425
426                         $RECORD_VALID=0;
427                         $MAILINGID_EXCEPT_ON = 1;
428                         $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   MAILING ID RECOVERY WAS NOT SUCCESSFUL"; &logEvent(\*F
    H_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
429
430                 }
431
432                 if($thisMailingID != -1) {
433
434                         $eventType = 'INFO'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   <<< MAILING ID RECOVERY SUCCESSFUL USING: $thisSubjectMa
    ilingIDToken -> $LAYOUT{$thisSubjectMailingIDToken}"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
435
436                 }
437         }
438
439         else {
440
441                 if($GDEBUG) { print "***DEBUG_  element1= $1 __ element2= $2 ___ element3= $3 ___ element4= $4\n"; sleep 1; }
442
443                 $element1 = $1; $element2 = $2; $element3 = $3; $element4 = $4;
444
445                 $thisCustNo             = ($element1 =~ /\S+?/) ? $element1 : -1;
446                 $thisListID             = ($element2 =~ /\S+?/) ? $element2 : -1;
447                 $thisMailingID          = ($element3 =~ /\S+?/) ? $element3 : -1;
448                 $thisDialogID           = ($element4 =~ /\S+?/) ? $element4 : -1;
449
450                 if ($thisCustNo == -1 && $thisListID == -1) {
451
452                         $LISTCID_EXCEPT_ON = 1;
453                         $MAILINGID_EXCEPT_ON = 1;
454                         $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   BAD MATCH:  CUSTNO, LISTID"; &logEvent(\*FH_log, $even
    tType, $eventDesc, $Guser, $LGDEBUG_FTN);
455                 }
456
457                 ## log/print INFO if cannot find mailingID
458
459                 if($thisMailingID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   NO MAILING ID MATCH"; &logEvent(\*FH_log, $eventTyp
    e, $eventDesc, $Guser, $LGDEBUG_FTN);}
460
461                 ## log/print INFO if cannot find dialogID
```

```
462
463             if($thisDialogID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   NO DIALOG ID MATCH"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
464         }
465
466 ## check for /multi || quoted BUY TOKEN (eg: "YES")
467
468         $buyTokenPosition       = 'none';
469         $thisItemsSelectedStack = undef;
470         @thisItemsSelected      = undef;
471
472         unless($ADD_CHANGE_ON) {
473
474                 ## check for buy tokens, eg: YES in SUBJECT, TOP, BOTTOM || letter items: ABC || A,B,C || A-C
475
476                         $buyTokenPosition = &detectBuyTokens($MULTI_BUY_ON,$Gsubj_buyToken,$thisSubject,$thisMailingID,$thisBodyTop,$thisBodyBottom,\%LAYOUT,\@topLines,\@bottomLines,$GDEBUG);
477
478                 ## check for no-match case and except
479
480                         if($buyTokenPosition eq 'none') { $eventRequiresFUP .= '.noTok' ; $FUPcount++; $eventType = 'ERR_FUP'; $warnCount++; $eventDesc = " \<".$thisRecordID."\>   SUBJECT, TOP, BOTTOM -> NO BUY TOKEN MATCH ($Gsubj_buyToken)"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN) ; }
481
482                 ## if match and MULTI_BUY_ON, prepare order items listing
483
484                         if($buyTokenPosition ne 'none' && $MULTI_BUY_ON) {
485
486                                 $thisItemsSelectedStack = &extractDedupItemsSelectedStack(@thisItemsSelected);
487
488                         }
489         }
490
491         ################
492         ## INNER LOOP ##
493         ################
494
495         %results                = {};
496         $countFields            = 0;
497         $nullCriticalFields     = 0;
498         $ADDRESS_ADEQUATE       = 1;
499
500 ## process body fields by key; keys == standard field names; values == this BODY's field names mapped thereon
501
502         foreach (@fieldKeys) {
503
504         $key = $_;
505
506         ## print "--DEBUG-- key = |$key| \n";
507
508                 $thisLayoutFieldName = uc($LAYOUT{$key});
509                 $thisLayoutFieldName =~ s/\s+$//;
510
511                 ###################################
512                 ###### MAIN MATCHING BLOCK ########
513                 ###################################
514
515                 $MATCH_DATA_ON = 0;
516                 $RULE = -1;
517
518                 $expectedMatch          = $thisLayoutFieldName.'\:.*?\[([^\[]+?)\]';
519                 $noBracketsMatch        = $thisLayoutFieldName.'\:.*?([^\012\015]+?)';
520
521                 CASE:  {
522
523                         TRY_EXPECTED_MATCH:     if($thisBody =~ /$expectedMatch/)               { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 1; last CASE; }
524                         TRY_NO_BRAKETS_MATCH:   if($thisBody =~ /$noBracketsMatch/)             { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 2; last CASE; }
```

```
525
526                 }
527
528 ## A MATCH WAS FOUND, THIS KEY ... push into results set
529
530                 if($MATCH_DATA_ON) {
531
532                         $countFields++;
533                         $matchElement  =~ s/[^\w\s\-\.\/\#\!\@]/$Gspace/g;
534                         $results{$key} =  &normalizeSpacing($matchElement);
535
536 ## print "--DEBUG-- result = |$results{$key}| \n";
537
538 ## build shipping and billing address blocks
539
540                         CASE: {
541
542                                 if($key =~ /u[a-z]_/) {
543
544                                         ## construct SHIPPING ADDRESSBLOCK for report
545
546                                         if ($key =~ /[hijklmq]\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
 }
547                                         if ($key =~ /[efgnop]\_/) { $thisShippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
 }
548
549                                 last CASE;
550                                 }
551
552                                 if($key =~ /[^u]_/) {
553
554                                         ## construct BILLING ADDRESSBLOCK for report
555
556                                         if ($key =~ /[hijklmq]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
}
557                                         if ($key =~ /[efgnop]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
}
558
559                                 last CASE;
560                                 }
561
562                         }
563
564                         ## check for null critical fields: any absence here will throw exception, unless CUSTNO
565
566 ## print "--debug-- |$LAYOUT{'criticalFields'}| \n";sleep 1;
567
568                         if ($key =~ /[$LAYOUT{'criticalFields'}]\_/ && $results{$key} !~ /\S+?/ && $thisCustNo < 1) {
569
570                                 $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
571                                 $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'.badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
tDesc = " \<".$thisRecordID."\>   CRITICAL STANDARD FIELD_VALUE IS NULL: \[$thisLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $L
GDEBUG_FTN);
572
573                         }
574
575                 }
576
577 ## ALL MATCHES FAILED, THIS KEY
578
579                 if(!$MATCH_DATA_ON) {
580
581                         $results{$key} = undef;
582                         $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   STANDARD FIELD_NAME=FIELD_VALUE PAIR UNPARSABLE: \[$th
isLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
583
584                 # check to see is unparsable field is a critical field
585                 # critical fields -> from CF file: any absence here will throw exception
586
```

```
587                         if ($key =~ /[$LAYOUT{'criticalFields'}]\_/) {
588
589                                 $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
590                                 $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'.badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   UNPARSABLE STANDARD FIELD_NAME=FIELD_VALUE PAIR IS CRITICAL: \[$thisLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
591                         }
592                 }
593
594         }
595
596         ######################
597         #### END INNER LOOP ###
598         ######################
599
600 ######################
601 ### SUMMARY RULES  ###
602 ######################
603
604 ## throw EX_ LISTNO/CUSTNO error exception if address is not OK && !phone, depending on CID_LID_REQUIRED from cf file
605
606         if($LAYOUT{'CID_LID_REQUIRED'}) {
607
608                  if($LISTCID_EXCEPT_ON) {
609
610                         $RECORD_VALID = 0;
611                         $eventRequiresFUP .= (($eventRequiresFUP =~ /chkCLID/)?'':'.chkCLID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   EITHER CID || LID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
612
613                 }
614
615         }
616
617         if(!$LAYOUT{'CID_LID_REQUIRED'}) {
618
619                 if($LISTCID_EXCEPT_ON && !($ADDRESS_ADEQUATE && $results{'r\_phone'} !~ /\S+?/)) { $RECORD_VALID = 0; }
620
621         }
622
623 ## throw EX_ MAILINGID exception if cf file says it's required
624
625         if($LAYOUT{'MAILINGID_REQUIRED'}) {
626
627                  if($MAILINGID_EXCEPT_ON) {
628
629                         $RECORD_VALID = 0;
630                         $eventRequiresFUP .= (($eventRequiresFUP =~ /chkMID/)?'':'.chkMID'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
631
632                 }
633
634         }
635
636 ## throw INFO if less than number required
637
638         if(($countFields < $numberRequiredFields)  && $thisCustNo < 1) {
639
640                 $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   (OVERALL) NOT ALL STANDARD FIELDS FOUND: $countFields ($numberRequiredFields)"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
641         }
642
643 ## tag duplicates for FUP
644
645 if($dupsMap{$mainCount} =~ /\#\#\#.+\#\#\#/) { $eventRequiresFUP .= (($eventRequiresFUP =~ /dupAd/)?'':'.dupAd'); $FUPcount++; }
646
647 ## catch thisBodyTop and thisBodyBottom blank yet address still extracted adequately (eg happens when cut & paste responses between the body delimiters)
648
649 if($RECORD_VALID && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/) {
650
```

```
651         $RECORD_VALID = 0;
652         $eventType = 'EX_ERR'; $eventRequiresFUP .= '.chkBody' ; $FUPcount++; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   CUSTOMER RESPONSE (Top, Bottom) NOT FOUND WHEN EXPECTED"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
653
654 }
655
656 ###############################################################
657 #### RECORD INVALID: Attempt recovery from mailing database ####
658 ###############################################################
659
660 $RECORD_RECOVERED = 0;
661
662 if(!$RECORD_VALID && !$ADD_CHANGE_ON) {
663
664         $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   >>> BAD RECORD: ATTEMPTING RECOVERY-LOOKUP USING \"$thisFromAddress\", $LAYOUT{'thisMailingDSN'} -> $LAYOUT{'thisMailingTable'}"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
665
666         %recoveredResults = undef;
667         %recoveredResults = &querySELECT_DB(\%LAYOUT,'EMAIL',$thisFromAddress,$LGDEBUG_FTN);
668
669 ## foreach(keys(%recoveredResults)) { print "=debug== key = $_ ____ value = $recoveredResults{$_} \n"; }
670
671         CASE: {
672
673                 if($recoveredResults{'_exit_'} < 1) {
674
675                         $RECORD_VALID = 0;
676                         $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB') ; $FUPcount++; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   MAILING TABLE RECOVERY-LOOKUP FAILED ON \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
677
678                 last CASE;
679                 }
680
681                 if($recoveredResults{'_exit_'} == 1) {
682
683                         $RECORD_RECOVERED = 1;
684                         $RECORD_VALID = 1;
685                         $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /goodMDB/)?'':'.goodMDB'); $FUPcount++; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   <<< MAILING TABLE RECOVERY-LOOKUP SUCCESSFUL USING \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
686
687                         ## construct $thisMDBbillingAddress
688
689                         foreach $key(sort(keys(%MDBLAYOUT))) {
690
691                                 $value = $MDBLAYOUT{$key};
692
693                         ## print "**DEBUG key = $key ___ value = $value \n";
694
695                                 if ($key =~ /[hijklmq]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$value}.$Gpipe):''; }
696                                 if ($key =~ /[efgnop]\_/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$value}.$Gspace):''; }
697
698                         }
699
700                         ## restore mailingID and CID, if blank
701
702 ## print "**DEBUG mailingID = $recoveredResults{'ED_MAILINGID'} __ custNo = $recoveredResults{'CID'} __ current custno = |$thisCustNo|\n";
703
704                         if($thisMailingID !~ /\S+?/ || $thisMailingID == -1)    { $thisMailingID        = $recoveredResults{'ED_MAILINGID'}; }
705                         if($thisCustNo !~ /\S+?/ || $thisCustNo == -1)          { $thisCustNo           = $recoveredResults{'CID'}; }
706
707                 last CASE;
708                 }
709
710         }
711
```

```
712 }
713
714 #### add an empty flag where SHIPPING ADDRESS is returned blank
715
716 if($thisShippingAddressBlock !~ /\S+?/) { $thisShippingAddressBlock = '{ SAME }'; }
717 if($thisMDBbillingAddressBlock !~ /\S+?/) { $thisMDBbillingAddressBlock = '{ N/A }'; }
718
719 #### tag address block exceptions with diagnostic ques
720
721 if(!$RECORD_VALID) {
722
723         CASE: {
724
725                 if($thisBillingAddressBlock !~ /\S+?/ && ($thisBodyTop =~ /\S+?/ || $thisBodyBottom =~ /\S+?/))         { $thisBillingAddressBlock = '=NO_
    VALID_ADD/INSPECT='; last CASE;}
726                 if($thisBillingAddressBlock !~ /\S+?/ && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/)          { $thisBillingAddressBlock = '{ BO
    DY_EMPTY/INSPECT }'; last CASE; }
727                 if($thisBillingAddressBlock =~ /\S+?/ && !$ADDRESS_ADEQUATE)                                          { $thisBillingAddressBlock = 'CRIT
    _FLD_NUL/INSPECT ->|'.$thisBillingAddressBlock.'|'; last CASE; }
728
729                 $thisBillingAddressBlock = '=UNKNOWN_ERR/INSPECT=|'.$thisBillingAddressBlock; last CASE;
730
731         }
732
733 $eventRequiresFUP = ($eventRequiresFUP !~ /\S+?/) ? '.inspect' : $eventRequiresFUP;
734 }
735
736 ########################################
737 ## write valid record data to outfiles ##
738 ########################################
739
740 $ED_ORDER_VALUE = 1;
741 $EXCEPTION_TYPE = 'OK';
742 $UPDATE_SUCCESSFUL = 0;
743 $THIS_KEY_FIELD = undef;
744 $THIS_KEY_FIELD_VALUE = undef;
745
746         if($RECORD_VALID) {
747
748                 unless($ADD_CHANGE_ON) {
749
750                         CASE: {
751
752                                 if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisFromAddress,$THIS_KEY_FIELD = "EMAIL","ED_ORDER",$ED_ORDER_VALUE,
    "ED_MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; };
753                                 if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisCustNo,$THIS_KEY_FIELD = "CID","ED_ORDER",$ED_ORDER_VALUE,"ED_MAI
    LACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN))        { $UPDATE_SUCCESSFUL=1; last CASE; };
754                                 if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisListID,$THIS_KEY_FIELD = "ED_LID","ED_ORDER",$ED_ORDER_VALUE,"ED_
    MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN))     { $UPDATE_SUCCESSFUL=1; last CASE; };
755
756                         }
757
758                 }
759
760                 if(!$UPDATE_SUCCESSFUL) {
761
762                         $RECORD_VALID = 0;
763                         $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badUpMDB/)?'':'.badUpMDB') ; $FUPcount++; $excpCount++; $event
    Desc = " \<".$thisRecordID."\>   (MANUAL UPDATE REQUIRED) MAILING TABLE UPDATE FAILED ON ALL KEYS - LAST TRY: \"$THIS_KEY_FIELD -> $THIS_KEY_FIELD_VALUE\"
    "; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
764
765                 }
766
767         }
768
769         if(!$RECORD_VALUE && !$UPDATE_SUCCESSFUL) {
770
771                 $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /noUpMDB/)?'':'.noUpMDB') ; $FUPcount++; $excpCount++; $eventDesc = " \
    <".$thisRecordID."\>   INVALID RECORD: MAILING TABLE NOT UPDATED"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
```

```
772
773         }
774
775 #### Buy tokens not found and not ADDRESS_CHANGE; move to EXC report
776
777         unless($ADDRESS_CHANGE) { if($buyTokenPosition eq 'none') { $RECORD_VALID = 0; $ED_ORDER_VALUE = 2; } }
778
779         if($RECORD_VALID) {
780
781                 $reportLineCount++;
782
783                 ## update MAILING TABLE
784
785
786                 ## write out files
787
788                 &writeRecord(\*FH_out,$GfieldDelimiter,($thisKey=&getNextDBkey_return($GoutFileDBkeysFileName,$LGDEBUG_FTN)),\%GoutFileRecord,'EXTENDED',
    $LGDEBUG_FTN);
789                 &writeRecord(\*FH_report,$GfieldDelimiter,($thisKey=&getNextDBkey_return($GreportFileDBkeysFileName,$LGDEBUG_FTN)),\%GreportFileRecord,'NO
    RMAL', $LGDEBUG_FTN);
790
791                 ## include ALL BODYs in the EXC_BODY file for potential review
792
793                 if($eventRequiresFUP =~ /\S+?/) { $EXCEPTION_TYPE = 'SOFT-EXC'; }
794
795                         &writeRecord(\*FH_excBODYReport,$GfieldDelimiter,$thisBODYlogKey=&getNextDBkey_return($GexceptionsBODYDBkeysFileName,$LGDEBUG_FTN)
    ,\%GexceptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);
796
797         $outFileCount++;$reportFileCount++;
798         }
799
800
801 ##############################################
802 ## write exception record data to outfiles ##
803 ##############################################
804
805         if(!$RECORD_VALID) {
806
807                 $EXCEPTION_TYPE = 'HARD-EXC';
808
809                 &writeRecord(\*FH_exc,$GfieldDelimiter,$thisKey=&getNextDBkey_return($GexceptionsDBkeysFileName,$LGDEBUG_FTN),\%GoutFileRecord,'EXTENDED',
    $LGDEBUG_FTN);
810                 &writeRecord(\*FH_excReport,$GfieldDelimiter,$thisKey=&getNextDBkey_return($GexcReportDBkeysFileName,$LGDEBUG_FTN),\%GreportFileRecord,'NO
    RMAL',$LGDEBUG_FTN);
811                 &writeRecord(\*FH_excBODYReport,$GfieldDelimiter,$thisBODYlogKey=&getNextDBkey_return($GexceptionsBODYDBkeysFileName,$LGDEBUG_FTN),\%Gexce
    ptionsBODYFileRecord,'BODY', $LGDEBUG_FTN);
812
813         $exceptionsFileCount++; $excReportFileCount++;
814         }
815
816 print "\n";
817 }
818
819 ########################
820 #### end main loop ####
821 ########################
822
823 ## print STDOUT exit state
824
825 $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
826
827 print "\n** [procOrder version $Grevision] Done!  $totalRecords records processed by $Guser/$GcfPath for client: $GclientName **\n\n";
828 print "SUMMARY REPORT for: ".$dateNow."\n\n";
829 print "FILES WRITTEN \n";
830 printf  "\n%10d",$reportFileCount;
831         print  " records written to REPORT FILE: $GreportFile";
832 printf  "\n%10d",$excReportFileCount;
833         print  " records written to EXC_REPORT FILE: $GexcReportFile";
834 printf  "\n%10d",$outFileCount;
```

```
835         print  " records written to OUTPUT FILE (all fields): $GoutFile";
836 printf  "\n%10d",$exceptionsFileCount;
837         print  " records written to EXCEPTIONS FILE (all fields): $GexceptionsFile";
838 print "\n\nLOGGING\n\n";
839 print "All rule failure events for all time logged in detail to: $GactivityLogFileName\n";
840 printf  "\n%10d",$totalRecords;
841 print " BODYs for this session are keyed, categorized in: $GexcBODYReportFile, for review in an editor\n";
842
843 $eventType = '_x_INFO'; $eventDesc = "procOrder R.$Grevision EXIT OK - USER: $Guser/$GcfPath CLIENT: $GclientName  opDSN ($totalRecords): $LAYOUT{'orderPr
    ocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thisMailingTable'}  ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent(\*
    FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
844
845 close(FH_out); close(FH_exc); close(FH_log); close(FH_report); close(FH_excReport);close(FH_excBODYReport);
846
847 exit;
848
849 ################################################
850 ################### SUBS #######################
851 ################################################
852
853 #
854 # ---------------------------------------------------------------------
855 #
856
857 sub alphaOrdered {
858
859 my(@singleElementArray) = @_;
860
861 my $alpha       = 0;
862 my $i           = 0;
863
864 my $base = $singleElementArray[0];
865
866 for($i = 1; $i <= $#singleElementArray; $i++) {
867
868         $ordBase = ord($base); $ordArr = ord($singleElementArray[$i]);
869
870 ##      print "==debug=$i= base = |$base| ordBase= $ordBase ___ singleElementArray = $singleElementArray[$i] ordArr = $ordArr \n";
871
872         if($ordBase > $ordArr) { return $alpha; }
873         $base = $singleElementArray[$i];
874
875 }
876
877 $alpha = 1;
878
879 return $alpha;
880
881 }
882
883 #
884 # ---------------------------------------------------------------------
885 #
886 sub canonicalize {
887
888 my($inRecord) = @_;
889 my($elementsIndex, $fieldsIndex, $rec, $buf, $field) = undef;
890 my(@field) = undef;
891 my $space = ' ';
892 my $dashEscape = $space.'_d_esc_'.$space;
893
894 $inRecord = uc($inRecord);
895
896 my @allFields = undef;
897
898 @allFields = split(/$GfieldDelimiter/,$inRecord);
899 my $fieldsIndex = undef;
900
901 for($fieldsIndex=0; $fieldsIndex <= $#allFields; $fieldsIndex++) {
```

```
902
903         if ($fieldsIndex != $emailFieldPos && $fieldsIndex != $fromAddressFieldPos) {
904
905                 if( $fieldsIndex ==  $companyFieldPos ||  $fieldsIndex == $lastNameFieldPos || $fieldsIndex == $firstNameFieldPos || $fieldsIndex == $titl
    eFieldPos || $fieldsIndex == $countryFieldPos || $fieldsIndex == $departmentFieldPos  || $fieldsIndex == $cityFieldPos)  { $allFields[$fieldsIndex] =~ s/\
    -/$dashEscape/g; }
906
907                 $allFields[$fieldsIndex] =~ s/\s+$//;                           # trailing space
908                 $allFields[$fieldsIndex] =~ s/^\s+//;                           # leading space
909                 $allFields[$fieldsIndex] =~ s/[\"\`\',\\\/\(\)]/$space/g;       # extraneous chars
910                 $allFields[$fieldsIndex] =~ s/\s{2,}/$space/g;                  # more than one separating space
911
912         }
913                 elsif ($fieldsIndex == $emailFieldPos || $fieldsIndex != $fromAddressFieldPos) { $allFields[$fieldsIndex] =~ s/$space//g; }
914
915         @field = split($space,$allFields[$fieldsIndex]);
916
917         $field = undef;
918
919         for($elementsIndex = 0; $elementsIndex <= $#field; $elementsIndex++) {
920
921                 CONVERT_TITLE_CASE: {
922
923                 $leaveUpper = 1;
924
925                         if($fieldsIndex == $postalCodeFieldPos) { last CONVERT_TITLE_CASE; }
926                         if($headerFields[$fieldsIndex] !~ /\w*IX\w*$/ && $field[$elementsIndex] =~ /^[^AEIOU\.][^AEIOU\.][^AEIOU\.]?$/) { last CONVERT_TIT
    LE_CASE; }
927                         if($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.[A-Z]?\.?$/)   { last CONVERT_TITLE_CASE; }
928
929                         unless($field[$elementsIndex] =~ /(AND)|(OR)|(NEW)|(UND)|(AT)|(CO)|(LTD)|(INC)|(OF)|(THE)|(RD)/) {
930
931                                 if($fieldsIndex == $firstNameFieldPos && ($field[$elementsIndex] =~ /^[A-Z][A-Z]$/ || $field[$elementsIndex] =~ /^[A-Z]\.?
    [A-Z]\.[A-Z]?$/)) { last CONVERT_TITLE_CASE; }
932                                 if(($fieldsIndex == $IXFieldPos ||  $fieldsIndex == $companyFieldPos)  && ($field[$elementsIndex] =~ /^[A-Z]\.?[A-Z]\.?[A-
    Z]$/ || $field[$elementsIndex] =~ /^[I][IV][I]*$/ || $field[$elementsIndex] =~ /^[^JS]\.?[A-Z]\.?$/ )) { last CONVERT_TITLE_CASE; }
933                                 if($fieldsIndex == $titleFieldPos && $field[$elementsIndex] =~ /^[ARHMGIVUCS][EMSQ\&TRAFIOPV][PADO\.]*$/ && $field[$elemen
    tsIndex] !~ /[MD][RS]/)     { last CONVERT_TITLE_CASE; }
934                                 if($fieldsIndex == $countryFieldPos && ($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]\.?[A-Z]?\.?$/ || $field[$elementsIndex] =
    ~ /^[A-Z][A-Z][A-Z]?$/) ) { last CONVERT_TITLE_CASE; }
935
936                         }
937
938                         unless($fieldsIndex == $stateFieldPos && length($field[$elementsIndex])>5) { if($fieldsIndex == $stateFieldPos || $fieldsIndex ==
    $postalCodeFieldPos)   { last CONVERT_TITLE_CASE; } }
939
940                 $leaveUpper = 0;
941
942
943                         $buf = lc($field[$elementsIndex]);
944
945                         if($fieldsIndex != $emailFieldPos && $fieldsIndex != $fromAddressFieldPos)  { $buf = ucfirst($buf); last CONVERT_TITLE_CASE; }
946                 }
947
948         if($leaveUpper) { $buf = $field[$elementsIndex]; }
949
950         $field .= ($buf.$space);
951         }
952
953 $rec .= ($field.$GfieldDelimiter);
954 }
955
956 chop($rec);
957
958 $rec =~ s/$dashEscape/\-/g;
959
960 return $rec;
961 }
962
```

```
963 #
964 # ------------------------------------------------------------------------------------------------
965 #
966
967 sub checkDupsReturnMap {
968
969 my($ptrAOHDB,$GDEBUG)    = @_;
970 my(@DB)                  = @$ptrAOHDB;
971
972 ## prepare map of duplicates on FromAddress
973
974 my %dupsMap              = undef;
975 my $j                    = undef;
976
977 for $j (0 .. $#DB) {
978
979         $dupsMap{$j} = $DB[$j]{'FromAddress'}
980
981 }
982
983 my @mirrorDupsMap        = undef;
984 @mirrorDupsMap           = values(%dupsMap);
985 my $key                  = undef;
986 my $value                = undef;
987 my @arr                  = undef;
988 my $number               = undef;
989
990 foreach $key(keys %dupsMap) {
991
992         $value           = $dupsMap{$key};
993         next if($value !~ /\S+?/);
994
995         @arr             = undef;
996         @arr             = grep /$value/, @mirrorDupsMap;
997         $number          = scalar(@arr);
998
999         if($number > 1) {
1000
1001                 $eventType      = 'INFO'; $eventDesc = "DUPLICATE FromAddress ITEM DETECTED ON IMPORT & FLAGGED: ($number|$key|$value)"; &logEvent(\*FH_lo
    g, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1002                 $dupsMap{$key}  = ('###'.$number.'### '.$dupsMap{$key});
1003
1004         ## print $dupsMap{$key}."\n";
1005
1006         }
1007         ## print "** key=".$key." *** value=".$dupsMap{$key}."___occurrences=".$number."\n";
1008 }
1009
1010 return %dupsMap
1011 }
1012
1013 #
1014 # ------------------------------------------------------------------------
1015 #
1016
1017 sub checkMultiBuy_Extract {
1018
1019 my($thisLine,$thisMailingID,$pos,$ptrLAYOUT,$GDEBUG) = @_;
1020 my($exit) = 0;
1021
1022 my(%LAYOUT)                     = %$ptrLAYOUT;
1023 my(@rangeStack)                 = undef;
1024 my(@singleStack)                = undef;
1025 my($countRangeMatches)          = 0;
1026 my($repeat)                     = 1;
1027 my($match)                      = undef;
1028 my($replacementEscChar)         = '%';
1029 my($preProcRemoveChar)          = '^';
1030 my($elementSeparator)           = ' ';
```

```
1031
1032 ## PREPARE THE LINE FOR MATCHING
1033
1034 $thisLine                        = uc($thisLine);
1035
1036 ## RUN RULESET FROM RULES CF FILE
1037
1038 foreach(@GPreProcRules) { eval; }
1039
1040 my $thisPreExtractionLine        = $thisLine;
1041 my @testSingleStack              = undef;
1042 my $maxASCII                     = 0;
1043
1044 ## EXTRACTION LOOP
1045
1046 while($repeat) {
1047
1048 my $RULE = 0;
1049 my $match = undef;
1050
1051 ##print "\n\n--debug-- maxASCII = $maxASCII -- thisLine IS CURRENTLY |$thisLine| \n\n";
1052
1053         MATCH_CASE: {
1054
1055                 SINGLE_CASE: {
1056
1057 ##                      if($thisLine =~ /^([A-Z])$/)                                                    { $match = $1; $RULE = -1;  last MATCH_CASE; }
1058                         if($thisLine =~ /^([A-Z])[^A-Z\^\%\@\-]{1}/)                                    { $match = $1; $RULE = -2;  last MATCH_CASE; }
1059                         if($thisLine =~ /[^A-Z\^\%\@\-]([A-Z])[^A-Z\%\^\@\-]{1}[^A-Z\^\%\@\-]*/)        { $match = $1; $RULE = -3;  last MATCH_CASE; }
1060
1061                 }
1062
1063                 RANGE_CASE: {
1064
1065                         if($thisLine =~ /^([A-Z]\-[A-Z])[^A-Z\^\%\@]{1}/)                               { $match = $1; $RULE = 1; last MATCH_CASE; }
1066                         if($thisLine =~ /[^A-Z\%\^\@]{1}([A-Z]\-[A-Z])[^A-Z\^\%\@]{1}/)                 { $match = $1; $RULE = 2; last MATCH_CASE; }
1067                         ## if($thisLine =~ /([A-Z][A-Z][A-Z]*)[^A-Z\^\%]{1}/)                           { $match = $1; $RULE = 3; last MATCH_CASE; }
1068                         ## if($thisLine =~ /[^A-Z]{1}([A-Z][A-Z][A-Z]*)/)                               { $match = $1; $RULE = 4; last MATCH_CASE; }
1069                 }
1070
1071
1072
1073         $repeat = 0;
1074         }
1075
1076         $thisLine =~ s/$match/$replacementEscChar/g;
1077
1078         $thisCheckedMailingID = ($thisMailingID!=0) ? $thisMailingID : -1;
1079
1080                 ##print "\n- debug - ELEMENT CONSIDERED with break = $RULE |$match| \n";
1081                 if($GDEBUG) { print "--debug-- BEGIN RULES -- RANGE = |$LAYOUT{$thisCheckedMailingID}| from $thisCheckedMailingID \n"; }
1082
1083 ## BEGIN RULES TO SEE WHAT WE GOT ##
1084
1085         $PUSH = 1;
1086
1087         if($RULE > 0) {
1088
1089                 my(@arr) = undef;
1090                 my $tempMatch = $match;
1091                 $tempMatch =~ s/\-//g;
1092                 @arr = split(//,$tempMatch);
1093
1094                 if(!&alphaOrdered(@arr)) {
1095
1096                         ## print "-- debug -- REJECT - not alpha match |$match| in line |$thisPreExtractionLine| \n";
1097                         $PUSH = 0;
1098
1099                 }
```

```
1100
1101            }
1102
1103            #if($RULE > 0 && ord($arr[0]) <= $maxASCII) {
1104
1105            #       print "-- debug -- REJECT - range case |$match| out of alpha order with current stack!\n";
1106            #       if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1107            #       $PUSH = 0;
1108
1109            #}
1110
1111            if(defined($match) && $match !~ /[$LAYOUT{$thisCheckedMailingID}]/) {
1112
1113                    ## print "--debug-- REJECT - not in RANGE = |$LAYOUT{$thisCheckedMailingID}| match |$match| in line |$thisPreExtractionLine|  \n" ;
1114                    if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1115                    $PUSH = 0;
1116
1117            }
1118
1119            if($PUSH && $RULE > 0 && $match =~ /\S+?/) {
1120
1121 ##                 push(@rangeStack,  ('<'.$RULE.'> |'.$match)); $exit = 1;
1122                    $match =~ s/\-/ thru /g;
1123                    push(@rangeStack, $match);
1124                    $exit=1;
1125            }
1126
1127            if($PUSH && $RULE < 0 && $match =~ /\S+?/) {
1128
1129                    push(@testSingleStack,$match);
1130
1131                    unless(ord($match) <= $maxASCII) {
1132
1133 ##                         push(@singleStack, ('<'.$RULE.'> |'.$match));
1134                            push(@singleStack,$match);
1135                            $exit=1;
1136
1137                    }
1138                            if(ord($match)>$maxASCII) {$maxASCII = ord($match); }
1139            }
1140
1141                    if($GDEBUG) { if(!defined($match)) {print "--debug-- match not defined! in line= |$thisPreExtractionLine|\n"; }}
1142 }
1143
1144 ## consolidate extractions, pre report
1145
1146 foreach(@rangeStack)    { unless($_ !~ /\S+?/) { push(@thisItemsSelected,$_); }}
1147 foreach(@singleStack)   { unless($_ !~ /\S+?/) { push(@thisItemsSelected,$_); }}
1148
1149 ## SPECIAL POST EXTRACTION CASE RULES
1150
1151         my $FIND_BUY_ALL = 0;
1152
1153         foreach(@GBuyAllIndicators) { s/[\n\r]//g; next if($_ !~ /\S+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_BUY_ALL = 1; } }
1154
1155         if ($FIND_BUY_ALL) {
1156
1157                 if($exit) {
1158
1159                         my $t = '.vfyTok';
1160                         $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS ALL-ITEM SELECTION DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1161
1162                 }
1163
1164                 push(@thisItemsSelected," ALL ($thisMailingID) "); $exit=1;
1165         }
1166
```

```
1167        ## DETECT DIFFERENCE IN EXTRACTION STACK AND MAX EXTRACTION STACK
1168
1169        if ($exit && (scalar(@singleStack) != scalar(@testSingleStack)) ) {
1170
1171                my $t = '.vfyTok';
1172                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t:  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
    = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING / INSPECT IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType
    , $eventDesc, $Guser, $LGDEBUG_FTN);
1173
1174        }
1175
1176        my $FIND_ADD_CHANGE = 0;
1177
1178        foreach(@GAddressChangeIndicators) { s/[\n\r]//g; next if($_ !~ /\S+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_ADD_CHANGE = 1; } }
1179
1180        if ($FIND_ADD_CHANGE) {
1181
1182                my $t = '.chkAC';
1183                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
     = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType,
     $eventDesc, $Guser, $LGDEBUG_FTN);
1184
1185        }
1186
1187        my $FIND_QUESTION = 0;
1188
1189        if($thisPreExtractionLine =~ /\?\s/) { $FIND_QUESTION = 1; }
1190
1191        if ($FIND_QUESTION) {
1192
1193                my $t = '.chkQst';
1194                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
     = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE OR CARE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
    entType, $eventDesc, $Guser, $LGDEBUG_FTN);
1195
1196        }
1197
1198        my $FIND_AI_ONLY = 0;
1199
1200        if ($exit && scalar(@singleStack) == 2 && scalar(@testSingleStack) == 2) {
1201
1202        ## $x = scalar(@singleStack);
1203        ## $y = scalar(@testSingleStack);
1204
1205        ##print "-- debug -- AI TEST ___ singleStackLen = $x __ testSingleStackLen = $y \n";
1206
1207                if(grep /(A)|(I)/, @singleStack) { $FIND_AI_ONLY = 1;}
1208
1209        }
1210
1211        if ($FIND_AI_ONLY) {
1212
1213                my $t = '.vfyTok';
1214                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
     = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: \"A\" or \"I\" ONLY DETECTED"; &logEvent(\*FH_log, $eventType, $eventD
    esc, $Guser, $LGDEBUG_FTN);
1215
1216        }
1217
1218        my $LIKELY_PARSE_ERRORS = 0;
1219
1220        if ($exit && $thisPreExtractionLine =~ /[\[\]]+/) { $LIKELY_PARSE_ERRORS = 1; }
1221
1222        if ($LIKELY_PARSE_ERRORS) {
1223
1224                my $t = '.vfyTok';
1225                $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t :  $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
     = " \<".$thisRecordID."\>   checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: BRACKETS FOUND IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
    entType, $eventDesc, $Guser, $LGDEBUG_FTN);
```

```
1226
1227         }
1228
1229
1230         ##print "\n-$thisRecordID- debug singleStack -- \n";
1231         ##print @singleStack;
1232         ##print "\n-- debug testSingleStack -- \n";
1233         ##print @testSingleStack;
1234         ##print "\n-$thisRecordID--------------------------\n";
1235
1236
1237 return $exit;
1238 }
1239
1240 #
1241 # ---------------------------------------------------------------------
1242 #
1243
1244 sub detectBuyTokens {
1245
1246 my($MULTI_BUY_ON,$Gsubj_buyToken,$thisSubject,$thisMailingID,$thisBodyTop,$thisBodyBottom,$ptrHASHLAYOUT,$ptrSCALARtopLines,$ptrSCALARbottomLines,$GDEBUG)
     = @_;
1247
1248 my %LAYOUT               = %$ptrHASHLAYOUT;
1249 my @topLines             = @$ptrSCALARtopLines;
1250 my @bottomLines          = @$ptrSCALARbottomLines;
1251 my $buyTokenPosition     = 'none';
1252 $thisBuyStatus           = undef;
1253 @thisItemsSelected       = undef;
1254 $thisItemsSelectedStack = undef;
1255
1256 my $exit = 0;
1257
1258 ## print "--debug-- buyToken= |$Gsubj_buyToken|\n";
1259
1260 FIND_BUY_TOKENS: {
1261
1262 ##
1263 ## &checkMultiBuy_Extract returns success on a good match/extraction;
1264 ## @thisItemsSelected is created/populated at this time as well
1265 ##
1266
1267 ## check $thisSubject
1268
1269 ## print "\n\n==========debug-- SUBJECT = |$thisSubject| \n";
1270
1271         if(!$MULTI_BUY_ON && $thisSubject =~ /$Gsubj_buyToken/) {
1272
1273                 $buyTokenPosition = ' S '; $thisBuyStatus = ('<'.$buyTokenPosition.'> |'.$thisSubject.'|');
1274
1275 ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1276
1277         last FIND_BUY_TOKENS;
1278         }
1279
1280         if($MULTI_BUY_ON) {
1281
1282                 if(&checkMultiBuy_Extract($thisSubject,$thisMailingID,' S ',\%LAYOUT,$GDEBUG)) {
1283
1284                         $buyTokenPosition = ' S '; $thisBuyStatus = '<'.$buyTokenPosition.'>';
1285
1286                 last FIND_BUY_TOKENS;
1287                 }
1288
1289         }
1290
1291 ## check @topLines
1292 ## print "\n\n==========debug-- thisBodyTop = |$thisBodyTop| \n";
1293
```

```
1294         if(!$MULTI_BUY_ON) {
1295
1296                 foreach(@topLines) {
1297
1298                         $thisTopLine = $_;
1299
1300                         if($thisTopLine =~ /$Gsubj_buyToken/) {
1301
1302                                 $buyTokenPosition = ' T ';
1303                                 $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisTopLine;
1304
1305 ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1306
1307                         }
1308
1309                 }
1310
1311         last FIND_BUY_TOKENS;
1312         }
1313
1314
1315         if($MULTI_BUY_ON) {
1316
1317                 if(&checkMultiBuy_Extract($thisBodyTop,$thisMailingID,' T ',\%LAYOUT,$GDEBUG)) {
1318
1319                         $exit =1;
1320
1321                         $buyTokenPosition = ' T ';
1322                         $thisBuyStatus = '<'.$buyTokenPosition.'> ';
1323
1324                 last FIND_BUY_TOKENS;
1325                 }
1326
1327         }
1328
1329
1330 ## check @bottomLines
1331
1332 ## print "\n\n==========debug-- thisBodyBottom = |$thisBodyBottom| \n";
1333
1334         if(!$MULTI_BUY_ON) {
1335
1336                 foreach(@bottomLines) {
1337
1338                         $thisBottomLine = $_;
1339
1340                         if($thisBottomLine =~ /$Gsubj_buyToken/) {
1341
1342                                 $buyTokenPosition = ' B ';
1343                                 $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisBottomLine;
1344
1345 ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1346
1347                         }
1348
1349                 }
1350
1351         last FIND_BUY_TOKENS;
1352         }
1353
1354         if($MULTI_BUY_ON) {
1355
1356                 if(&checkMultiBuy_Extract($thisBodyBottom,$thisMailingID,' B ',\%LAYOUT,$GDEBUG)) {
1357
1358                         $exit =1;
1359
1360                         $buyTokenPosition = ' B ';
1361                         $thisBuyStatus = '<'.$buyTokenPosition.'> ';
1362
```

```
1363                    last FIND_BUY_TOKENS;
1364                    }
1365
1366            }
1367
1368 }
1369
1370 ## print "--debug-- ==== $i ====== buyTokenPosition= |$buyTokenPosition|\n";
1371
1372 return $buyTokenPosition;
1373 }
1374
1375 #
1376 # ------------------------------------------------------------------------
1377 #
1378
1379 sub extractDedupItemsSelectedStack {
1380
1381 my(@thisItemsSelected) = @_;
1382
1383 my $thisItemsSelectedStack = undef;
1384 my $thisItemsSelectedStackSeparator = ',';
1385 my $i = undef;
1386
1387 for($i = 0; $i <= $#thisItemsSelected; $i++) {
1388
1389         my $item = $thisItemsSelected[$i]; # all items must be normalized!
1390
1391         next if($item !~ /\S+?/);
1392
1393         $item=&normalizeSpacing($item);
1394
1395         $thisItemsSelectedStack .= ($item.$thisItemsSelectedStackSeparator);
1396
1397 }
1398
1399 $thisItemsSelectedStack =~ s/\,\s*$//;
1400
1401 return  $thisItemsSelectedStack;
1402 }
1403
1404 #
1405 # ------------------------------------------------------------------------
1406 #
1407
1408 sub getDateStamp {
1409
1410 # get standardized date stamp (UNIX, WINNT)
1411 # returns a date stamp
1412
1413 my($Gplatform,$LGDEBUG_FTN) = @_;
1414
1415 if(!defined($Gplatform)) { $Gplatform = 'WIN'; }
1416 my $GTIME       = '';
1417 my $day         =  undef;
1418 my $val         =  undef;
1419
1420 if ($Gplatform ne 'WIN') { $val = `date '+%y%m%d%H%M%S-%a'`; }
1421         else {
1422                 $val = `date /t`;
1423                 chop($val);
1424                 $val =~ /(\w+)\s+(\d+)\/(\d+)\/(\d{2})(\d{2})/;
1425                 $val = $5.$2.$3; $day = $1;
1426         }
1427
1428 if ($Gplatform eq 'WIN') {
1429
1430                 $GTIME = `echo '' | time`;
1431                 $GTIME =~ /\:\s*(\d{1,2})\:(\d{2})\:(\d{2})\.(\d{2})/;
```

```
1432                    my $leadingDigit = $1;
1433                    if (length($leadingDigit) == 1) { $leadingDigit = ('0'.$leadingDigit); }
1434                    $val .= $leadingDigit.$2.$3.$4.'-'.$day;
1435            }
1436
1437 if($LGDEBUG_FTN) { print "DEBUG__gdate=".$val."\n"; }
1438
1439 return $val;
1440
1441 }
1442
1443 #
1444 # ------------------------------------------------------------------------
1445 #
1446
1447 sub getFieldKeys {
1448
1449 my($ptrLayout) = @_;
1450 my(%LAYOUT) = %$ptrLayout;
1451
1452 ## prepare and count required fields from body data layout for check against body
1453
1454 my(@fieldKeys)  = undef;
1455 my($k)          = undef;
1456
1457 ##$f = scalar(@fieldKeys); print "--debug-- scalarStartFtn=$f \n";
1458
1459 foreach $k(sort keys %LAYOUT) {
1460
1461         next if($k !~ /^[a-z]{1,2}_/);  ## these elements are not fieldnames but are other CF parameters (name \t value)
1462
1463         ## print "--debug-- k= |$k|\n";
1464
1465         if($k =~ /\S+?/) { push(@fieldKeys,$k); }
1466 }
1467
1468 shift @fieldKeys;
1469 ## $f = scalar(@fieldKeys); print "--debug-- scalarEndFtn=$f \n";
1470
1471 return @fieldKeys;
1472 }
1473
1474 # ------------------------------------------------------------------------------------------------
1475
1476 sub getKeyPos_Outfile {
1477
1478 my($ptrHASHrecordTemplate) = @_;
1479
1480 my(%HASHrecordTemplate) = %$ptrHASHrecordTemplate;
1481
1482 my @record                      = undef;
1483 $salutationFieldPos     = -1;
1484 $IDFieldPos             = -1;
1485 $LIDFieldPos            = -1;
1486 $MailingIDFieldPos      = -1;
1487 $firstNameFieldPos      = -1;
1488 $lastNameFieldPos       = -1;
1489 $IXFieldPos             = -1;
1490 $titleFieldPos          = -1;
1491 $stateFieldPos          = -1;
1492 $countryFieldPos        = -1;
1493 $postalCodeFieldPos     = -1;
1494 $emailFieldPos          = -1;
1495 $companyFieldPos        = -1;
1496 $addressFieldPos        = -1;
1497 $departmentFieldPos     = -1;
1498 $cityFieldPos           = -1;
1499 $fromAddressFieldPos    = -1;
1500
```

```
1501 my $i                    = 0;
1502
1503 @record = sort(keys(%HASHrecordTemplate));
1504
1505 ##print "---DEBUG---"; print @record; sleep 4;
1506
1507 for($i=0; $i <= $#record; $i++) {
1508
1509         $record[$i] = uc($record[$i]);
1510
1511         CASE: {
1512
1513                 if ($record[$i] =~ /FROM\_ADDRESS/)                                                 { $fromAddressFieldPos  = $i; last CASE; }
1514                 if ($record[$i] =~ /(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNET_AD)|(INTERNETAD)/)   { $emailFieldPos        = $i; last CASE; }
1515                 if ($record[$i] =~ /(COUNTRY)|(CTRY)/)                                              { $countryFieldPos      = $i; last CASE; }
1516                 if ($record[$i] =~ /(ZIP)|(POSTAL)/)                                                { $postalCodeFieldPos   = $i; last CASE; }
1517                 if ($record[$i] =~ /(INST)|(COMPANY)|(ORGANIZATION)/)                               { $companyFieldPos      = $i; last CASE; }
1518                 if ($record[$i] =~ /DEPARTMENT/)                                                    { $departmentFieldPos   = $i; last CASE; }
1519                 if ($record[$i] =~ /(ADDR)|(ADDRESS)/)                                              { $addressFieldPos      = $i; last CASE; }
1520                 if ($record[$i] =~ /CITY/)                                                          { $cityFieldPos         = $i; last CASE; }
1521                 if ($record[$i] =~ /(CUSTNO)/)                                                      { $IDFieldPos           = $i; last CASE; }
1522                 if ($record[$i] =~ /(LISTID)/)                                                      { $LIDFieldPos          = $i; last CASE; }
1523                 if ($record[$i] =~ /(PRIORITY)|(MAILING)/)                                          { $MailingIDFieldPos    = $i; last CASE; }
1524                 if ($record[$i] =~ /(.*F\w+NAME.*)|(FIRST)/)                                        { $firstNameFieldPos    = $i; last CASE; }
1525                 if ($record[$i] =~ /(.*L\w+NAME.*)|(LAST)/)                                         { $lastNameFieldPos     = $i; last CASE; }
1526                 if ($record[$i] =~ /^.*IX.*$/)                                                      { $IXFieldPos           = $i; last CASE; }
1527                 if ($record[$i] =~ /TITLE/)                                                         { $titleFieldPos        = $i; last CASE; }
1528                 if ($record[$i] =~ /STATE/ || $record[$i] =~ /^\_C$/)                               { $stateFieldPos        = $i; last CASE; }
1529         }
1530 }
1531
1532 if ($firstNameFieldPos  == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\+F\\w\+NAME\/ was not found!\n"; }
1533 if ($lastNameFieldPos   == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\+L\\w\+NAME\/ was not found!\n"; }
1534 if ($firstNameFieldPos  == -1)          { print STDOUT "[HEADER] WARNING: \/\\w\*IX\\w\*NAME\/ was not found!\n"; }
1535 if ($titleFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/TITLE\/ was not found!\n"; }
1536 if ($stateFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/STATE\/ was not found!\n"; }
1537 if ($postalCodeFieldPos == -1)          { print STDOUT "[HEADER] WARNING: \/ZIP|POSTAL\/ was not found!\n"; }
1538 if ($countryFieldPos    == -1)          { print STDOUT "[HEADER] WARNING: \/(COUNTRY)|(CTRY)\/ was not found!\n"; }
1539 if ($emailFieldPos      == -1)          { print STDOUT "[HEADER] WARNING: \/(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNET_AD)|(INTERNETAD)\/ was not found!\n
     "; }
1540
1541 }
1542
1543
1544 #
1545 # ------------------------------------------------------------------
1546 #
1547
1548 sub getNextDBkey_return {
1549
1550 my($keyFileName,$LGDEBUG_FTN) = @_;
1551
1552 $LOCK_SH = 1;
1553 $LOCK_EX = 2;
1554 $LOCK_NB = 4;
1555 $LOCK_UN = 8;
1556
1557 $POS_BOF = 0;
1558 $POS_CUR = 1;
1559 $POS_EOF = 2;
1560
1561 $/ = undef;
1562 $startKeys = 0;
1563
1564 if(! -s $keyFileName) { `echo $startKeys >> $keyFileName`; }
1565
1566 OPEN_DB: if (!open(KEYS, "+< $keyFileName")) {$exit = 0; $eventType = 'FAIL'; $eventDesc = "getNextDBkey_return::Cannot open: \[$keyFileName \]"; &logEven
     t(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
1567
```

```
1568 flock(KEYS, $LOCK_EX);
1569
1570         my($count)      = <KEYS>;
1571         $nextKey        = $count+1;
1572         seek(KEYS,0,$POS_BOF);
1573         print KEYS $nextKey;
1574
1575 flock(KEYS, $LOCK_UN);
1576 close(KEYS);
1577
1578 return $nextKey;
1579 }
1580
1581 #
1582 # ------------------------------------------------------------------------
1583 #
1584
1585 sub HASHloadFieldMap {
1586
1587 my($fileName,$LGDEBUG_FTN) = @_;
1588
1589 if (!open(FH, "$fileName")) { $eventType = 'FAIL'; $eventDesc = "HASHloadFieldMap::Cannot open: \[$keyFileName \]"; &logEvent(\*FH_log, $eventType, $event
     Desc, $Guser, $LGDEBUG_FTN);}
1590
1591 $/ = "\n";
1592
1593 my(@fieldMapTemp) = <FH>;
1594 my($key) = undef;
1595 my($value) = undef;
1596 my(%fieldMap) = undef;
1597
1598 close(FH);
1599
1600         foreach (@fieldMapTemp) {
1601
1602                 my $item = $_;
1603
1604                 next if(($item =~ /\/\//) || ($item !~ /\S+?/)); # comments
1605
1606                 ($key, $value) = split(/[\t\,]/,$item);
1607
1608                 $value =~ s/[\n\r]//g;
1609
1610                 $fieldMap{$key} = $value;
1611
1612         }
1613
1614 return %fieldMap;
1615 }
1616
1617 #
1618 # ------------------------------------------------------------------------
1619 #
1620
1621 sub loadFile_SCALAR {
1622
1623 my($fileName) = @_;
1624
1625 if (!open(FILE, "$fileName")) {$eventType = 'FAIL'; $eventDesc = "loadArray::Cannot open fileName: \[$fileName\]"; &logEvent(\*FH_log, $eventType, $eventD
     esc, $Guser, $LGDEBUG_FTN);}
1626
1627 $/ = "\n";
1628
1629 my(@all) = undef;
1630
1631 while(<FILE>) {
1632
1633         my $buf = $_;
1634         next if($buf !~ /\S+?/ || $buf =~ /^\/\//);
```

```
1635         push(@all, $buf);
1636
1637 }
1638
1639 return @all;
1640 }
1641
1642 #
1643 # ---------------------------------------------------------------------
1644 #
1645
1646 sub logEvent {
1647
1648 my($FH,$eventType,$eventDesc,$Guser,$LGDEBUG_FTN) = @_;
1649
1650 my $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
1651 # print event to STDOUT
1652
1653 if($eventType eq 'ABORT') {
1654
1655         $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$Guser $GclientName $eventT
    ype $eventDesc"`;
1656         die "Error! Cannot log! ".$eventType."\t".$eventDesc."\n" ;
1657 }
1658
1659 if($eventType !~ /_x_/) { print $eventType."\t".$eventDesc."\n"; }
1660
1661         else { $eventType =~ s/_x_//; }
1662
1663 # write event to activity log
1664
1665         &writeRecord(\*FH_log,$GfieldDelimiter,$thisLogKey=&getNextDBkey_return($GactivityLogKeysFileName,$LGDEBUG_FTN),\%GactivityLogRecord,'',$LGDEBUG_F
    TN);
1666         if($eventType eq 'FAIL') {
1667
1668                 $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procOrder - Critical Error"  -m "$Guser $GclientName
     $eventType $eventDesc"`;
1669                 exit;
1670
1671         }
1672 }
1673
1674 #
1675 # ---------------------------------------------------------------------
1676 #
1677
1678 sub normalizeSpacing {
1679
1680 my($dirty) = @_;
1681
1682 $dirty =~ s/\s{2,}/$space/g; $dirty =~ s/\s+$//; $dirty =~ s/^\s+//;
1683 $clean = $dirty;
1684
1685 return $clean
1686 }
1687
1688 #
1689 # ---------------------------------------------------------------------
1690 #
1691
1692 sub queryUPDATE_DB {
1693
1694 my($ptrLAYOUT,$thisQueryKey, $thisQueryFieldName, $fieldName, $fieldValue, $fieldName1, $fieldValue1, $RECORD_RECOVERED, $LGDEBUG_FTN) = @_;
1695
1696 ##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";
1697
1698 my(%LAYOUT) = %$ptrLAYOUT;
1699 my $RS = undef;
1700 my $Conn = undef;
```

```
1701 my $Errors = undef;
1702 my @fieldNames = undef;
1703 my $PID = -1;
1704 my $recCount = 0;
1705 my $EXIT = 1;
1706
1707 # construct/open DSN
1708
1709 my $Conn = undef;
1710
1711 $Conn = new Win32::OLE("ADODB.Connection");
1712 $Conn->Open($LAYOUT{'thisMailingDSN'},"","");
1713
1714 my(%DB) = undef;
1715
1716 # Load data into recordset
1717
1718         my $mailingTable = $LAYOUT{'thisMailingTable'};
1719         my %foo = undef;
1720
1721 # look to see if update record exists, $thisQueryKey
1722
1723         unless($RECORD_RECOVERED) {
1724
1725                 my $RECORD_EXISTS = 1;
1726                 %foo = &querySELECT_DB(\%LAYOUT,$thisQueryFieldName,$thisQueryKey,$LGDEBUG_FTN);
1727                 if($foo{'_exit_'} < 1) { $RECORD_EXISTS = 0; $EXIT = 0; return $EXIT; }
1728         }
1729
1730         $thisQuery = "UPDATE $mailingTable SET $fieldName = \'$fieldValue\', $fieldName1 = \'$fieldValue1\'  WHERE $thisQueryFieldName = \'$thisQueryKey\'
     ";
1731
1732         $RS = $Conn->Execute($thisQuery);
1733
1734 ##%foo = %$RS;
1735 ##foreach(keys(%foo)) { print "---key = $_ __ value = $foo{$_} \n"; }
1736
1737         if(!$RS) {
1738
1739                 $Errors = $Conn->Errors();
1740                 print "Errors:\n";
1741                 foreach $error (keys %$Errors) {
1742                         print $error->{Description}, "\n";
1743                 }
1744                 $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB'); $eventDesc = " \<".$thisRecordID."\>   ($LAYO
     UT{'thisMailingDSN'}) queryUPDATE_DB :: Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1745
1746         $EXIT = 0;
1747         return $EXIT;
1748         }
1749
1750         $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) queryUPDATE_DB :: UPDATE SUCCESSFUL: \[$thisQuery\]"; &log
     Event(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1751
1752 return $EXIT;
1753 }
1754
1755 #
1756 # ---------------------------------------------------------------------
1757 #
1758
1759 sub querySELECT_DB {
1760
1761 my($ptrLAYOUT,$thisQueryFieldName,$thisQueryKey,$LGDEBUG_FTN) = @_;
1762
1763 ##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";
1764
1765 my(%LAYOUT) = %$ptrLAYOUT;
1766 my $RS = undef;
```

```
1767 my $Conn = undef;
1768 my $Errors = undef;
1769 my @fieldNames = undef;
1770 my $PID = -1;
1771 my $recCount = 0;
1772
1773 # construct/open DSN
1774
1775 my $Conn = undef;
1776
1777 $Conn = new Win32::OLE("ADODB.Connection");
1778 $Conn->Open($LAYOUT{'thisMailingDSN'},"","");
1779
1780 my(%DB) = undef;
1781 $DB{'_exit_'} = 1;
1782
1783 # Load data into recordset
1784
1785 my $mailingTable = $LAYOUT{'thisMailingTable'};
1786
1787 $thisQuery = "SELECT * FROM $mailingTable WHERE $thisQueryFieldName = \'$thisQueryKey\'";
1788
1789 $RS = $Conn->Execute($thisQuery);
1790
1791 if(!$RS) {
1792
1793         $Errors = $Conn->Errors();
1794         print "Errors:\n";
1795         foreach $error (keys %$Errors) {
1796                 print $error->{Description}, "\n";
1797         }
1798         $eventType = 'FAIL'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingDSN'}) querySELECT_DB :: Cannot create RS: \[$thisQuery\]"; &logE
     vent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1799 }
1800
1801 my $i = undef;
1802
1803 for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1804
1805 my $RESULT_FLAG = 1;
1806
1807 if($RS->EOF) { $RESULT_FLAG = 0; }
1808 $RS -> MoveNext;
1809
1810 if(!$RS->EOF) { $RESULT_FLAG = -1; }
1811 $RS -> MovePrevious;
1812
1813 if($RESULT_FLAG < 1) {
1814
1815         (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB'); $FUPcount++; $eventType = 'ERR_FUP'; $eventDesc = " \<".$thisRecordID."\>   ($LAYOUT{'thisMailingD
     SN'}) querySELECT_DB :: Bad Lookup ($RESULT_FLAG): \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1816
1817         $DB{'_exit_'} = $RESULT_FLAG;
1818
1819 return %DB;
1820 }
1821
1822 while ( !$RS->EOF ) {
1823
1824         for($i=0; $i < $RS ->Fields->Count; $i++) {
1825
1826                 $DB{$fieldNames[$i]}= uc($RS->Fields->Item($i)->value);
1827         }
1828
1829         $RS->MoveNext; $recCount++;
1830 }
1831
1832 $Conn->Close;
1833
```

```
1834 return %DB;
1835 }
1836
1837 #
1838 # --------------------------------------------------------------------
1839 #
1840
1841 sub read_DSNtoAOH {
1842
1843 my($DSN,$LGDEBUG_FTN) = @_;
1844
1845 ##print "** DSN = $DSN \n";
1846
1847 $RS = undef;
1848 $Conn = undef;
1849 $Errors = undef;
1850 @fieldNames = undef;
1851 $PID = -1;
1852 $recCount = 0;
1853
1854 # construct/open DSN
1855
1856 $Conn = new Win32::OLE("ADODB.Connection");
1857 $Conn->Open($DSN,"","");
1858 my(@FILE) = undef;
1859
1860 # Load data into recordset
1861
1862 $thisQuery = "SELECT * FROM Email";
1863
1864 $RS = $Conn->Execute($thisQuery);
1865
1866 if(!$RS) {
1867         $Errors = $Conn->Errors();
1868         print "Errors:\n";
1869         foreach $error (keys %$Errors) {
1870                 print $error->{Description}, "\n";
1871         }
1872         $eventType = 'FAIL'; $eventDesc = "read_DSNToAOH::Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_
    FTN);
1873 }
1874
1875 my $i = undef;
1876
1877 for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1878
1879 while ( !$RS->EOF ) {
1880
1881         for($i=0; $i < $RS ->Fields->Count; $i++) {
1882
1883          $FILE[$recCount]{$fieldNames[$i]}= $RS->Fields->Item($i)->value;
1884
1885         }
1886
1887         $RS->MoveNext; $recCount++; if ($recCount % 20 == 0) { print STDOUT "+"; }
1888 }
1889
1890 $Conn->Close;
1891
1892 return @FILE;
1893 }
1894
1895 #
1896 # --------------------------------------------------------------------
1897 #
1898
1899 sub removeTabs {
1900
1901 my($tabs) = @_;
```

```
1902
1903 $tabs = s/\t//g;
1904
1905 return $tabless
1906
1907 }
1908
1909 #
1910 # ---------------------------------------------------------------------
1911 #
1912
1913 sub subjectSynonymnLookup {
1914
1915 my($subject,$ptrHASHLAYOUT) = @_;
1916 my(%LAYOUT) = %$ptrHASHLAYOUT;
1917 my($thisMailingID) = -1;
1918
1919         $subject =~ /\s*([\=\*\+\-]{1,3})\s*/;
1920         $thisSubjectMailingIDToken = $1;
1921
1922         if(defined($LAYOUT{$thisSubjectMailingIDToken})) { $thisMailingID = $LAYOUT{$thisSubjectMailingIDToken}; }
1923
1924         ## print "-- debug -- ATTEMPT MAILING ID RECOVERY... TOKEN= |$thisSubjectMailingIDToken|... RESULT MAILINGID= |$thisMailingID|\n";
1925
1926 return $thisMailingID
1927 }
1928
1929 #
1930 # ---------------------------------------------------------------------
1931 #
1932
1933 sub writeRecord {
1934
1935 # takes FH by ref; has _e_ feature for values
1936
1937 my($FH,$GfieldDelimiter,$logKey,$ptrHASHrecord,$RECORD_TYPE, $LGDEBUG_FTN) = @_;
1938
1939 flock($FH, $LOCK_EX);
1940
1941 my $fileEmpty = 0;
1942 if(-z $FH) {$fileEmpty = 1;}
1943
1944 %HASHrecord = %$ptrHASHrecord;
1945
1946 $space = " ";
1947 $exit = 1;
1948
1949 $LOCK_SH = 1;
1950 $LOCK_EX = 2;
1951 $LOCK_NB = 4;
1952 $LOCK_UN = 8;
1953
1954 my $record = undef;
1955 my $header = undef;
1956 my $key = undef;
1957
1958 foreach $key(sort(keys(%HASHrecord))) {
1959
1960         $value  = $HASHrecord{$key};
1961
1962         if($fileEmpty) { $key =~ s/[a-z]{1,2}_//; $header .= (uc($key).$GfieldDelimiter); }
1963
1964         if($value =~ /^_e_(.{2,})$/) { $value = eval($1); }
1965
1966
1967         $record .= ($value.$GfieldDelimiter);
1968
1969 }
1970 chop($record);                                  # remove last field delimiter
```

```
1971
1972 if($fileEmpty) { chop($header); print $FH $header."\n"; }
1973 if($ADD_CHANGE_ON && $RECORD_TYPE ne 'BODY') { $record = &canonicalize($record); }
1974 print $FH $record."\n";
1975
1976 flock($FH, $LOCK_UN);
1977 }
1978
1979 __END__
1980
```